REMARKS

Claims 6, 18, and 24 have amended. No claims have been canceled. No new claims have been added. Claims 1-24 are pending.

Claims 1-24 stand rejected under 35 U.S.C. 112, second paragraph as being indefinite due to the use of the limitation "wherein the privileged function is executed as part of the same execution thread as the application." This rejection is respectfully traversed. More specifically, the Office Action states that:

1. The application at Fig. 3 illustrates a system "which switches to the privileged stack and privileged registers while, seemingly, saving the non-privileged thread's context"; and

2. A prior art publication not associated with the claimed invention, viz., U.S. Patent No. 6,175,916 (hereinafter the '916 patent or "Ginsberg") states in its background section that "[g]enerally, an execution thread comprises a sequence of processor instructions that execute in a single processor context. The particular elements of a thread's context vary depending upon the microprocessor being used. For purposes of the discussion herein, however, a thread's context always includes it private memory stack or stacks. Therefore, by definition, a single thread always uses the same private memory stack. Any time the processor context changes to a different memory stack, the processor is said to be executing a different thread."

The Office Action states that an apparently conflict between paragraphs (1) and (2) leads to the rejection under 35 U.S.C. 112, second paragraph. More specifically, the Office Action continues by stating that "[a]s best the examiner can determine, applicant's system switches from executing the user thread, to executing the privileged thread, and back to executing the user thread. This does not seem to be a situation where one of ordinary skill would recognize the privilege function being executed as part of the user thread; rather, the privileged function seems to operate as a separate thread."

It is respectfully asserted that the Office Action is in error. Fundamentally, the '916 patent defines a "thread" a manner suitable for its own purposes. This is permitted, as the principle that each patentee is entitled to be their own lexicographer is well settled. However, the limitations accorded to a "thread" in the '916 patent cannot be automatically imputed to any subsequent patent application. To do so would be to preclude any inventions which would improve upon the described subject matter by making such subject matter more flexible by removing one or more of such limitations, as in the present application.

Additionally, it should be noted that under many processor architectures, the user mode is associated with a first stack while the privileged mode is associated with a second stack. See., e.g., application at Fig. 2 (showing code and data memories divided into independent partitions for user and privileged modes). Thus, in many processor architectures, switching between user and privileged modes necessarily requires switching between stacks, and transitioning from user to privileged to user modes would require saving the user mode stack pointers to permit the transition back to user mode without a loss of information. For these reasons, the rejection under 35 U.S.C. 112, second paragraph, should be withdrawn.

Further, it should be noted that the definition of a "thread" in the '916 patent is incomplete. For example, the passage immediately after that quoted in paragraph 2 above, column 1, lines 46-58, describe the operation of a conventional scheduler, and in particular, imply that threads require scheduling for execution. Column 2, lines 16-31 further state that in a conventional system, a privilege mode task would be executed by a "system function" which is separately scheduled.

Yet, one important aspect of the present invention is the ability for one thread to switch from user mode to privilege mode and then back to user mode, without requiring the creation of a privileged mode thread, thereby avoiding the overhead associated with thread creation. See application at paragraph [0046]. The application therefore maintains consistency between what is described in its specification and the subject matter sought to be claimed. Accordingly, this provide an additional basis supporting the withdraw of the rejection under 35 U.S.C. 112, second paragraph.

Claims 1-5, 8-11, 13-16, 18-22 stand rejected under 35 U.S.C. 102(b) as being anticipated by Ginsberg (U.S. Patent No. 6,175,916, i.e., the '916 patent). Claims 7, 12, 17, and 24 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Ginsberg in view of applicant's disclosure. These rejections are respectfully traversed.

Claim 1 recites, *inter alia,* "transitioning to the privileged mode to execute the privileged function, wherein the privileged function is executed as part of the same thread of execution as the application."

Claim 6 has been rewritten as an independent claim has recites, *inter alia,* "transitioning to the privileged mode to execute the privileged function, wherein the transitioning comprises: switching to a privileged mode stack; and the privileged function is executed as part of the same thread of execution as the application."

Claim 8 recites, *inter alia,* "transition logic that operates to transition to the privileged mode to execute the privileged function, wherein the privileged function is executed as part of the same execution thread as the application."

Claim 13 recites, *inter alia,* "means for transitioning to the privileged mode to execute the privileged function, wherein the privileged function is executed as part of the same execution thread as the application."

Claim 18 recites, *inter alia,* "instructions for transitioning to the privileged mode to execute the privileged function, wherein the privileged function is executed as part of the same thread of execution as the application."

The Office Action states that Ginsberg discloses a system in which a user mode thread executes a privileged function by transitioning to a privileged mode which executes in the same thread as the user mode thread, and specifically identifies three passages for support, namely: column 5, lines 1-23; column 6, line 65 – column 7, line 20; and column 8, line 50 to column 10, line 17.

<u>1. column 5, lines 1-23</u>;

Column 5, lines 1-23 describe processor 41 of the Ginsberg, which is disclosed to be a conventional off the shelf microprocessor, such as the MIPS R3000 processor. This passage further discloses that the microprocessor supports privileged and user modes, and the use of fault handlers. However, contrary to the assertion in the Office Action, this passage fails to disclose or suggest any user mode application having a thread which transitions to a privileged mode such that "the privileged function is executed as part of the same thread of execution as the application," as recited in independent claim 1. Independent claims 6, 8, 13, and 18 recite similar limitations.

2. column 6, line 65 – column 7, line 20.

Column 6, line 65 – column 7, line 20 describe how Ginsberg utilizes a jump instruction to an invalid address to trigger a memory fault, causing its operating system to execute a call handler, which will recognize a call to a system function if the invalid memory address is an odd address. However, contrary to the assertion in the Office Action, this passage fails to disclose or suggest any user mode application having a thread which transitions to a privileged mode such that "the privileged function is executed as part of the same thread of execution as the application," as recited in independent claim 1. Independent claims 6, 8, 13, and 18 recite similar limitations.

3. column 8, line 50 to column 10, line 17.

Column 8, line 50 to column 10, line 17 is part of a larger passage starting from column 7, line 21 to column 10, line 28, describing Fig. 9. The extended passage describes how a user process 100 may call a system process 102 in the following manner. First, the user process 100 creates a memory fault. Column 8, line 49-53. Responsive to the memory fault, a fault handler is invoked. Column 8, lines 52-60. The fault handler identifies the memory address which created the memory fault, and maps that address to a system call, and then calls the identified system call. Significantly, the invocation of the fault handler means that execution has switched away from the thread in the user process 100 which caused the memory fault and to a different thread. The passage further describe how a call processing function is utilized to process memory addresses prior to the fault handler calling the identified system call in the system process 102. Significantly, Ginsberg discloses that the fault handler, the call processing function, and the system function itself are executed within the same execution thread. Column 9, 65-67. Because the fault handler is invoked by the processor in response to the memory fault caused by the user thread in the user process 100, the user thread must be a different thread than that of the fault handler, call processing function, and system function.

Thus, contrary to the suggestion made in the Office Action, Ginsberg fails to disclose or suggest any user mode application having a thread which transitions to a privileged mode such that "the privileged function is executed as part of the same thread of execution as the application," as recited in independent claim 1. Independent claims 6, 8, 13, and 18 recite similar limitations. In fact, not only does Ginsberg disclose using a separate thread (i.e., one comprising the fault handler, call processing function, and system function) to service the system call, the Ginsberg even executes the system call in a non-privileged mode. For example, column 9, lines 54 – 57 states the "[f]ault handler 104 switches to the system address space of the specified system function, restores the previous execution mode (normally non-privileged mode) and calls the system function."

Independent claim 6 further recites, "wherein the transitioning comprises: switching to a privileged mode stack." It should be noted that Ginsberg discloses that the calling process 101 does not switch stacks at column 9, lines 4-5 ("the thread that made the system call continues to executed with the same stack that was used by the calling process.")

Accordingly, independent claims 1, 6, 8, 13, and 18 are allowable. The depending claims are also believed to be allowable for at least the same reasons as the independent claims.

CONCLUSION

In light of the amendments contained herein, Applicants submit that the application is in condition for allowance, for which early action is requested.

Please charge any fees or overpayments that may be due with this response to Deposit Account No. 17-0026.

Respectfully submitted,

Dated: December 23, 2005

By: [illegible]

Christopher S. Chow
Reg. No. 46,493
(858) 845-3249

QUALCOMM Incorporated
Attn: Patent Department
5775 Morehouse Drive
San Diego, California 92121-1714
Telephone: (858) 658-5787
Facsimile: (858) 658-2502